COMPUTING SCIENCE

Efficient encoding of instructions of ARM Cortex M0+ processor

Alessandro de Gennaro and Paulius Stankaitis

TECHNICAL REPORT SERIES

No. CS-TR-1442 January 2015

TECHNICAL REPORT SERIES

No. CS-TR-1442 January, 2015

Efficient encoding of instructions of ARM Cortex M0+ processor

Alessandro de Gennaro, Paulius Stankaitis

Abstract

As the complexity of digital hardware grows steadily, so does the demand of high level modelling systems to abstract the low level implementation of synchronous and asynchronous digital circuits. Conditional partial order graphs representation provides a compact and solid approach to represent event-based systems, and together with Workcraft design tool, they represent an extremely efficient development environment to create any system. So far, the synthesis phase of the controller for managing the whole datapath structure lacked an efficient encoding association mechanism, as well as a friendly and customisable user interface to encode the Partial order graphs. The following paper aims at bridging this gap, providing a graphical user interface integrated in Workcraft, new algorithms for the generation of encodings for the Partial orders composing the system and the possibility to customise the final op-codes, if needed. All these features have been tested on the Instruction Set Architecture ARMv6-M, integrated into the ARM Cortex M0+ microprocessor showing quite a big improvement in terms of area of the final controller with respect to previous approaches.

Printed and published by Newcastle University,
Computing Science, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, England.

Bibliographical details

DE GENNARO, A., STANKAITIS, P.

Efficient encoding of instructions of ARM Cortex M0+ processor
[By] A. de Gennaro, P. Stankatis

Newcastle upon Tyne: Newcastle University: Computing Science, 2015.

(Newcastle University, Computing Science, Technical Report Series, No. CS-TR-1442)

Added entries

NEWCASTLE UNIVERSITY
Computing Science. Technical Report Series. CS-TR-1442

Abstract

As the complexity of digital hardware grows steadily, so does the demand of high level modelling systems to abstract the low level implementation of synchronous and asynchronous digital circuits. Conditional partial order graphs representation provides a compact and solid approach to represent event-based systems, and together with Workcraft design tool, they represent an extremely efficient development environment to create any system. So far, the synthesis phase of the controller for managing the whole datapath structure lacked an efficient encoding association mechanism, as well as a friendly and customisable user interface to encode the Partial order graphs. The following paper aims at bridging this gap, providing a graphical user interface integrated in Workcraft, new algorithms for the generation of encodings for the Partial orders composing the system and the possibility to customise the final op-codes, if needed. All these features have been tested on the Instruction Set Architecture ARMv6-M, integrated into the ARM Cortex M0+ microprocessor showing quite a big improvement in terms of area of the final controller with respect to previous approaches.

About the authors

Alessandro De Gennaro is a Research Assistant at Newcastle University and is involved in a project named "Asynchronous for Analogue electronics". Throughout this project he is going to deal with the design flow for designing asynchronous circuitry, by means of Petri nets, Signal Transitioning Graph and Conditional Partial Order Graph models. The main goal of the project is to develop tools for supporting verification, synthesis and mapping algorithms suitable for asynchronous circuits which need to communicate with analogue electronics modules (for power management such as buck controllers).

Paulius Stankaitis received his BEng in Electronic Engineering at Newcastle University. His final thesis, entitled "Algebraic Specifications of ARM Cortex M0+ Instruction set", was in the area of system modelling and verification. Currently, he is working as an RA at the School of Computing Science in the SafeCap+ project.

Suggested keywords

EFFICIENT INSTRUCTION ENCODING
CONDITIONAL PARTIAL ORDER GRAPHS

Efficient encoding of instructions of ARM Cortex M0+ processor

Alessandro de Gennaro[1], Paulius Stankaitis[2]

School of Electrical and Electronic Engineering[1], Newcastle University, United Kingdom
School of Computing Science[2], Newcastle University, United Kingdom
alessandro.de-gennaro@newcastle.ac.uk, paulius.stankaitis@newcastle.ac.uk

December 22, 2014

Abstract

As the complexity of digital hardware grows steadily, so does the demand of high level modelling systems to abstract the low level implementation of synchronous and asynchronous digital circuits. Conditional partial order graphs representation provides a compact and solid approach to represent event-based systems, and together with `Workcraft` *design tool, they represent an extremely efficient development environment to create any system. So far, the synthesis phase of the controller for managing the whole datapath structure lacked an efficient encoding association mechanism, as well as a friendly and customisable user interface to encode the Partial order graphs. The following paper aims at bridging this gap, providing a graphical user interface integrated in* `Workcraft`*, new algorithms for the generation of encodings for the Partial orders composing the system and the possibility to customise the final op-codes, if needed. All these features have been tested on the Instruction Set Architecture* `ARMv6-M`*, integrated into the* `ARM Cortex M0+` *microprocessor showing quite a big improvement in terms of area of the final controller with respect to previous approaches.*

I. Introduction

Over the years aggressive transistor's scaling lead to immense microprocessor performance improvement, for instance microprocessor running frequency has increased 100 times more than theoretically predicted. However, power consumption of chip has been increasing and became crucial progress constraint [4]. Moreover, as in recent decade mobile electronics became more and more affordable to customers, devices operation time became vital. Though, battery life has not increased significantly [5], thus other power optimization approaches are being researched. Reduction of power consumption through Instruction Set optimization approach has been known for a while. Nonetheless , it is still an active research area. In this field some important work has been done by μSystem Research Group at Newcastle University: introduction of formalism called, Conditional Partial Order Graph; new ISA design approach based on that model. Reduction of power limitation through ISA motivates to further explore this fairly recently introduced, though promising design approach.

II. Conditional Partial Order Graph Background

Conditional Partial Order Graph [1][6][7] is a quintuple H = (V, E, X, ρ, ϕ):

- **V** is a set of vertices which correspond to events (or atomic actions) in a modelled system.
- **E** $\subseteq VxV$ is a set of arcs representing dependencies between the events.
- Operational vector **X** is a set of Boolean variables. An opcode is an assignment $(x_1, x_2, ..., x_{|X|}) \in 0, 1^{|X|}$ of these variables. An opcode selects a particular partial order from those contained in the graph.
- $\boldsymbol{\rho} \in F(X)$ is a restriction function, where $F(X)$ is the set of all Boolean functions over variables in X. ρ defines the operational domain of the graph: X can be assigned only those opcodes $(x_1, x_2, ..., x_{|X|})$ which satisfy the restriction function, i.e. $\rho(x_1, x_2, ..., x_{|X|}) = 1$.
- Function $\boldsymbol{\phi} : (V \cup E) \to F(X)$ assigns a Boolean condition $\phi(z) \in F(X)$ to every vertex and arc $z \in V \cup E$ in the graph. Let us also define $\phi(z) \stackrel{df}{=} 0$ for $z \notin V \cup E$ for convenience.

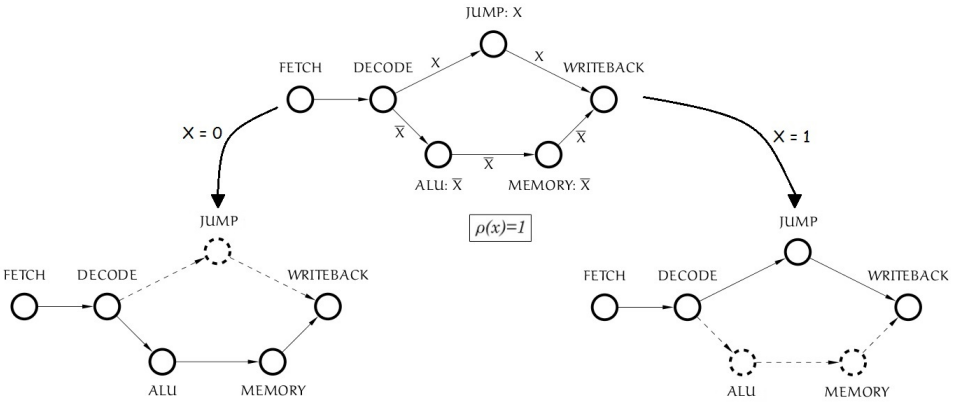

Figure 1: *Example of CPOG with 2 projections: $H|_{X=1}$ on the right, $H|_{X=0}$ on the left side*

This model is based on strict graphical representation, where each event is represented by a circle ◯, and each connection between vertices is named arc, depicted as an arrow $\rightarrow$. Both the previous elements are labelled with a predefined pattern composed by $vertex/arc$ name, followed by condition $\phi(v/e)$. Next to each graph, a further condition is present called $restrictionfunction$ (ρ), composed by $operationalvariables X$.

An Example of CPOG is shown in Figure 1 with two possible projections at the bottom-side. The purpose of the condition ϕ is to switch on/off vertices and edges, when the conditions on it are satisfied or not respectively. In this representation, dash edges and circles represent nodes and arrows switched off, in such a way not to affect the behaviour of that particular event class.

III. Encoding of different Partial Orders

The *encoding process* can be performed after that all the Partial Orders have been created and optimised, it consists in associating an unique op-code to each graph in order to be distinguished by the other ones. This is the first step of the synthesis phase, where the controller for managing the whole structure will be instantiated. In order to better describe such problem, it may be worth giving some definitions of the terms used along this section.

Op-code points out a variable length array of bits where each element can be a logic `0` or `1`, the following ones are examples of feasible op-codes $\{0010, 000000, 1110001, 0\}$. **Op-code ensemble** is defined as the the group of *op-codes* it is possible to use with a particular number of bits, for instance the *Op-code ensemble* of length 2 is composed by *op-codes*: $\{00, 01, 10, 11\}$ while the one of length 3 $\{000, 001, 010, 011, 100, 101, 110, 111\}$. An **encoding**, also referred to as **solution**, is a subset of an *op-code ensemble* where each *op-code* is associated to one and only one *Partial Order* graph. Finally, the **solution space** is represented as the group of all the possible *encodings* for a particular CPOG.

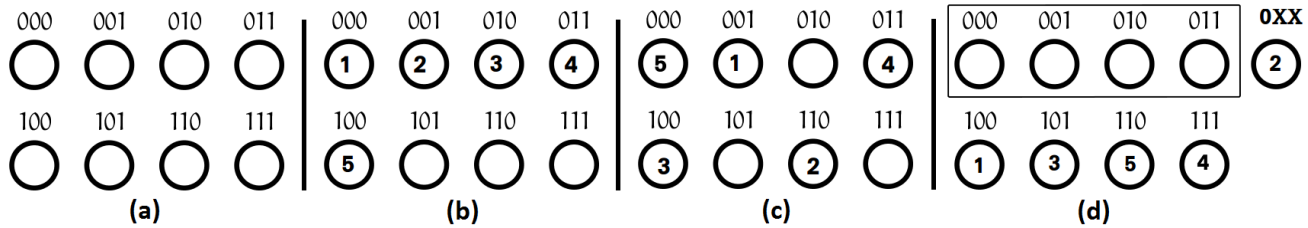

Figure 2: *Some examples of possible encodings for a 5 Partial orders model.*

On Figure 2-a are depicted the op-codes available to encode a CPOG that contains up to 8 *Partial Orders*. Each op-code is composed by 3 bits. While on Figure 2-b and 2-c, two possible solutions for what concerns a 5 graphs CPOG are showed. Designers may also exploit *Don't care* conditions when any particular op-codes don't cover any Partial orders, as showed on Figure 2-d. Here, the second Partial order is encoded by the op-code 0XX.

On the light of above, we want to demonstrate that the *solution space* might be really high depending on the

number of graphs to encode and on the size of the *op-code ensemble*. As our aim now is to make the reader understand the size of the problem, in order to compute the number of solutions globally available for a single CPOG we are going to neglect the encodings including *Don't cares* conditions in. The size of the *solution space* is of primarily importance because potentially, each *solution* might be a good *encoding* in terms of area and should be inspected then. As a consequence, the higher the number of solutions, the harder would be to seek the best possible encoding to select for a particular representation. This is why one of the main concern at this step is trying to reduce the size of the *solution space* as much as possible.

For instance, let us consider a very small *Conditional Partial Order Graph* representation, composed by 4 graphs only. As analysed in [7] and briefly above, there could be several approaches to encode various graphs in order to minimise the Boolean function in each vertex and edge. What we are going to take into account below will be *op-codes ensemble* with minimum length related to number of *Partial Orders*.

It means that in order to encode 4 different graphs, we need 2 bits (*Op-code ensemble* of length 2). Therefore, in order to get the minimum number of bits needed to encode an entire model we need to use formula 1, where m stands for the size of current *op-code ensemble*, assuming the graphs to be encoded with op-codes on minimum number of bits.

$$\#\{\mathcal{B}\} = \lceil \log_2(k) \rceil \tag{1}$$

In this case, number of *encodings* fits perfectly to the graphs to encode as with two bits it is possible to encode exactly four elements ($2^2 = 4$). Additionally, in order to compute the entire solution space of such instance we should take into account all the permutations of the *op-code ensemble*. Therefore, the total number of solutions we can select by sorting elements differently each time in a group of m different elements is represented on Formula 2 by $\#\{\mathcal{S}\}$, where k is the number of graphs to encode.

$$\#\{\mathcal{S}\} = \frac{m!}{(m-k)!} \qquad \Rightarrow \qquad \#\{\mathcal{S}\}' = \frac{(m-1)!}{(m-k)!} \qquad \Rightarrow \qquad i.e. \quad \frac{(4-1)!}{(4-4)!} = \frac{3!}{1} = 3! = 6 \tag{2}$$

Additionally, we managed to reduce the *solution space* more by fixing the first element for every possible solution. It is because each circuit may be derived by two different encodings (adding some inverter gates) complementary each other. Thus, by taking into account statement before, we modified $\#\{\mathcal{S}\}$ into $\#\{\mathcal{S}\}'$ (Formula 2). It is extremely beneficial for the size of the *solution space*. By analysing problem before the ensemble reduces from 24 to 6 as proved on Formula 2.

I. Heuristic function

After discussed about the number of solutions one should deal with, it is worthwhile mentioning that in the literature of the Conditional partial order Graphs, a quick way to heuristically evaluate the area of the final controller by looking at the encoding only was missed. As a direct consequence, in order to evaluate a solution in terms of area consumption, it should have gone through the logic synthesis and technology mapping steps; making the whole process of seeking the best possible solution between the ones available an extremely difficult task. This research, and in particular in this Section, therefore aims at presenting a heuristic function for an early evaluation of an encoding.

In order to describe the heuristic function just mentioned, let us first attempt to describe the differences between the Partial Order Graphs composing a single model algebraically. One might count how much a graph is different from another one, and set this result inside a matrix (defined as *Difference Matrix*). In particular, it is defined as a strict upper triangular matrix $[\mathcal{N} \times \mathcal{N}]$, where $\mathcal{N}$ is the number of graphs in a model, with all the entries on the main diagonal fixed to 0. Every row r is associated to a *PO*, as well as every column c. Each entry represents how much the Partial order r, is different from the c one. An example is depicted below on the left-side of Formula 3. In a similar way it is possible to store all the differences between the op-codes assigned in the $\mathcal{H}$amming $\mathcal{D}$istance matrix (right-side of Formula 3).

$$DM = \begin{pmatrix} 0 & 2 & 1 & 4 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad \mathcal{HD} = \begin{pmatrix} 0 & 1 & 1 & 2 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{3}$$

Key point to find the cost function is to encode the graphs in such a way that: the more two graphs are similar, the higher the priority to encode them with two similar op-codes, that is, with a minimum $\mathcal{H}$amming $\mathcal{D}$istance.

On the light of above, cost function I used to find minimum area is on Formula 4. $\mathcal{DM}_{i,j}$ represents entry of the matrix $\mathcal{DM}$ with $row = i$ and $column = j$, while $\mathcal{HD}_{i,j}$ represents $\mathcal{H}$amming $\mathcal{D}$istance between op-codes used to encode i and j *CPOGs*.

$$\mathcal{F} = \sum_{i,j \in \mathcal{N}}^{i \neq j} (\mathcal{DM}_{ij} - \mathcal{HD}_{ij})^2 \tag{4}$$

Intuitively, minimising $\mathcal{F}$ means encoding similar *Partial orders* with couple of op-codes with lower $\mathcal{HD}$. In order to demonstrate the applicability of such cost function, we tried to synthesise all the possible solutions of a representation composed by eight graphs (presented on [13], Figure 2.7) and map the controllers generated with a 90nm simple library composed by basic 2 input gates. The results taking into account area and the heuristic function are plotted on Figure 3-a where it is clearly present a strong correlation between the area of the final circuit and the cost stem by the encoding.

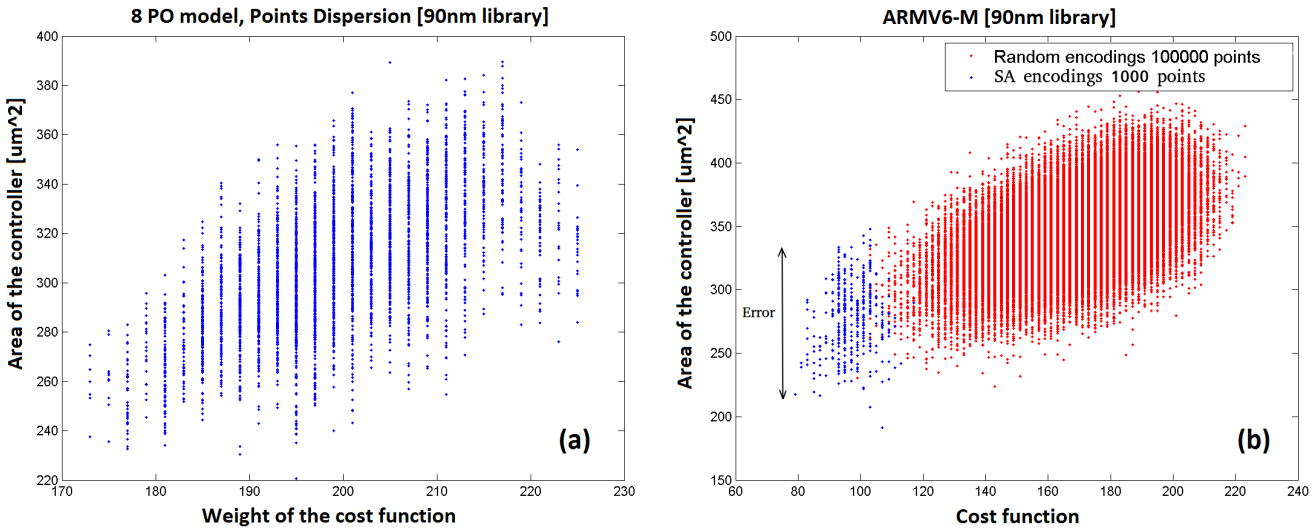

Figure 3: ***(a)****: Points dispersion of CPOG composed by 8 graphs;* ***(b)****: Comparison between Simulated Annealing and Random search techniques applied to ARMV6-M ISA model.*

II. Encoding generation

Exploiting the heuristic function presented previously, designers may achieve quite good results in terms of area of the final controller without going through the synthesis phase of the model. Last part to complement the best encoding association process is to present the encoding generation algorithms that have been developed.

For what concerns the *Encoding generation* process, designers have mostly two choices: trying to generate all the possible solutions and evaluate them by the function, or generating few solutions targeting a low cost of function. It is worth mentioning that, even though the former approach would guarantee a better result, it is not always applicable to all models as the number of solutions might explode in the number for the bigger representations. During this work, both the two possibilities are allowed to designer who might freely choose them in the tool developed and briefly presented in Section V. As the longer approach is pretty straightforward, the *clever* approach will be analysed hereinafter only, the interested reader is referred to [13] for further information. Briefly, the algorithm generates as many solutions as the designer wants trying to associate dynamically couple of op-codes that present a lower *Hamming distance* to more similar graphs. As some steps of the algorithm are decided randomly, for example when pairs of graphs or op-codes contain the same number of differences, the result is not guaranteed to be optimal, so an additional step is performed in order to optimised the solution found. Such a step is based on Simulated Annealing optimisation technique.

Hereinafter referred to also as *SA*, it was introduced by Kirkpatrick et al. in 1983. As they stated in their work: "The subject of combinatorial optimization [9] consists of a set of problems that are central to the disciplines of computer science and engineering. Research in this area aims at developing efficient techniques for finding

minimum or maximum values of a function of very many independent variables [10]."([8]). Commonly, it is named *cost function*, and in our case is the one on Formula 4. Several research were conducted exploiting such optimisation method, either in computer engineering field or in other subjects (i.e. [11], [12]). In our case we have used this approach swapping the op-codes associated to various Partial order attempting to minimise the cost function. The result, applied to the same model mentioned before composed by eight Partial orders are depicted on Table 1. The last row, which presents the generation algorithm described so in this Section shows up a quite good result in terms of area-time.

Table 1: *Comparison between encodings generations applied to a CPOG with eight Partial orders.*

Generation approach	**# Encoding inspected**	**Area** $[\mu m^2]$	**Time** $[s]$	**Error from exact**
Recursive search	5040 (all)	220,630	2105	0%
Random search	100	247,970	17	12,39%
Simulated annealing search	10	237,520	2	7,65%

The results of this Chapter will be applied to an Instruction Set Architecture of microprocessor in the next Chapter, in such a way to have an extremely sound case of study.

IV. Case of study: Instruction Set Architecture of ARM Cortex M0+

In order to further explore CPOG formalism and encoding process, a case study was completed with ARM Cortex M0+ processor's instructions as a study basis. Methodology for this work was introduced in [16] and was also followed throughout this research. First step required a derivation of partial orders and it was mainly done by analysing instruction set description. Although, before Partial orders can be constructed, datapath components need to be specified. Past work done by μSystem Research Group at Newcastle University obtained five datapath units [14], four were reused in this study, Table 2 describes each of them.

Table 2: *Datapath units that model instructions of ARM Cortex M0+ processor.*

Component	Description
ALU	Executes mathematical operations.
MAU	Access internal and external memory.
PCIU	Increments Program Counter (PC).
IFU	Provides opcodes to Instruction Register (IR).

Derivation of partial orders involved deep analysis of individual instruction, which was done using ARMv6-M technical manual [3]. Manual provides detailed information about every instruction including encoding, operation pseudo-code and function of instruction. Process was simplified due to ability to cluster instruction and give class a single PO. During derivation process it was noticed that in particular two aspects of instruction affect its graphical representation: *functional similarity* and *addressing mode*. Since, `ARM Cortex M0+` instructions can have two types of addressing modes to specify operands: *register and immediate addressing*. Obtained graphs shared this property in their representation. In particular, addressing mode affected operand fetching process part, instructions with *register type* operand were able to fetch next instruction concurrently with currents execution, while *immediate addressing* type instructions required to fetch constant in order to complete instruction.

Figure 4 shows two types of graphs derived in this case study. Both, represents nearly the same amount and functionality of instructions, although key difference, which make them different is addressing mode. As explained, graph on the left hand side does not require operand fetching, hence $PCIU \rightarrow IFU$ in this graph represents next instruction fetching, while the $PCIU \rightarrow IFU$ in the graph on the right operand fetching.

Functionality difference was an another aspect, which differentiated graphs. It's perhaps more intuitive than addressing mode differentiation, as one may understand various purpose instructions are executed in different manner, thus resulting in unique graph. However, instructions can be still categorized under their functionality. General groups are: arithmetic - operate, memory access, control and miscellaneous.

Specification of Instruction Set procedure extracted 11 different Partial orders, which covered in total 64 `ARM Cortex M0+` instructions. Majority of covered instructions were arithmetical and logical. However, not all

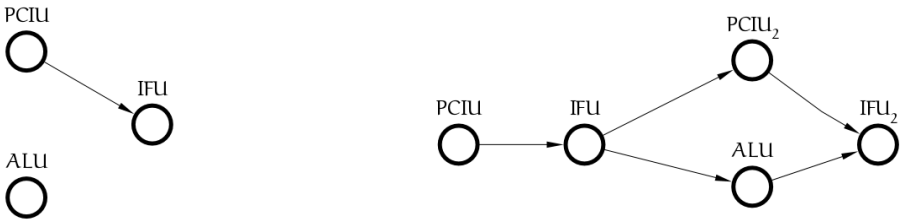

Figure 4: *Register addressing mode and immediate mode instructions*

processor instructions were covered. In particular, hint type instructions, which are associated with interrupts, microprocessor state mode. These instruction required more insightful analysis and due to project time constraints and very high level description of these instruction in [3] they were left out. It is reasonable to predict that effect these instructions could have had on size and complexity of CPOG and control circuit are insignificant.

I. Compositional results

After derivation of Partial orders was completed, research proceeded with the next step: encoding and generation of CPOGs. Although in this research three different algorithms have been developed and analysed that a designer may use to compose the CPOGs, it was decided to focus deeply on two approaches in particular: **Random** encoding generation and the **Simulated annealing** method, the most meaningful for this application. Those two algorithms indeed fit well to a CPOG with such a high number of graphs as the one described, and additionally designers may find these two techniques suitable to have a rough idea of the area this technique allows to save due to the wide difference of the two methods. Number of gates and area of controller were used as parameters to compare differently generated encodings.

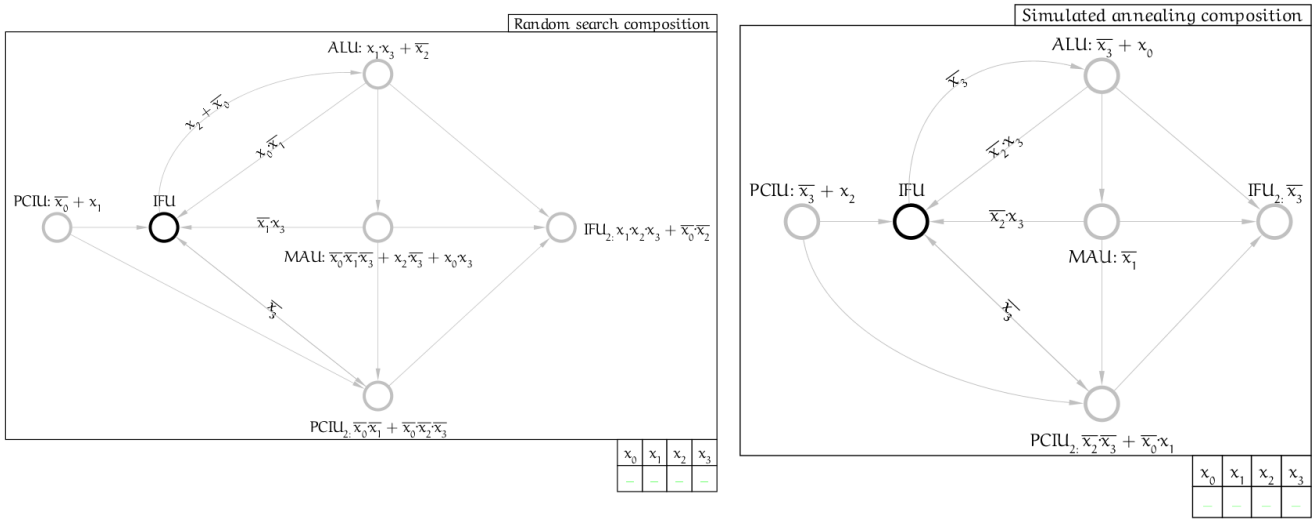

Figure 5: *CPOGs generated with Random and Simulated annealing encoding algorithms*

Before going on, let us put into a Table (3) the best results, in terms of area and number of gates, obtained with four different approaches applied to the Partial orders representing the `ARMV6-M` Instruction Set Architecture.

Table 3: *Controller size comparison for ARMv6-M ISA model.*

	Sequential	**Random**	**Exhaustive search** [17]	**Simulated annealing**
Area $[\mu m^2]$	335,47	226,53	204,80	191,090
N. Gates	50	36	29	26

The worst solution is the Sequential search, it generates the biggest controller. It is pareto-dominated by all the other techniques, but it is useful for giving a general idea about the whole area of the controller without using any of the optimisation techniques developed. Random approach whereas gives a quite good result even though hundred thousands of solutions have been examined taking quite a long time. Exhaustive search approach, described in [17], gives a very good result comparable to the Simulated annealing approach which in this case give the best result by analysing one thousand encodings only. For understanding the goodness of this approach, reader should take into account that, according to [13] (Figure 4.7), the size of the solution space for a representation composed by 11 POs is around 10^{10} possible encodings.

Now, let us compare the two heuristic techniques (random and simulated annealing approaches) that this paper presents and and let us show the soundness of the solutions that the Simulated annealing algorithm is able to find. As we mentioned before, we tried to generate an extremely high number of solutions with Random encoding, and a small one with Simulated annealing search. The result is surprisingly good, and showed on Figure 3-b.

Aim indeed was seeking few and sound encodings via the latter technique respectively. On the x-axis of the graph is present the cost function presented in SectionI, while on the y-axis the bare area [$\mu\ m^2$] is present. As we demonstrated, the lower the cost function of the solution, the higher is the probability to select a better encoding in terms of area, so the main goal is to have a heuristic returning encodings minimising the cost function. As one might observe indeed, even though the extremely high number of solutions generated randomly, the weight of the cost function related to each encoding is centred around 180, and rarely touches 110 ([13], Figure 6.7). The reason of such a bad result resides on the massive number of possibilities the solution space is composed by. Indeed, even though 100000 may seem a high number of solutions, it represents the 0,0009 % of solution space only, so it is extremely unlikely to find a really good point. On the other hand, Simulated annealing approach is able to centre the average of the encoding found on 100, touching 80 as well.

Only drawback depending on the heuristic cost function is related to the non linearity of the solutions found. The error the intrinsically affect the solution space constrains the tool to analyse a range of solutions centred on the minimum, as the best controller in terms of area might be present with a slightly higher cost of the function.

On the light of above it can be affirmed that one of the main goals of this research has been addressed. The composition of Partial Order graphs is now automated exploiting an algorithm which is also able to build the whole system in an optimal way, designers that use this model can now freely compose their own system quickly and easily through the SCENCO (SCEnario ENCOder) tool present in the Conditional Partial Order Graph plugin under Workcraft.

V. Conclusions and Future research directions

As was already demonstrated, *Conditional Partial Order Graphs* model provides a compact environment to represent a system that includes different behaviours, either synchronous or self-timed. As proved in Section IV additionally, CPOG tailors well to Instruction Set Architecture development, it may help designers to seek an optimal and efficient encoding as well as to automatically instantiate the decoding module. This work aims at filling the gap in the synthesis part allowing designers to choose the encoding for the representation under development via different generation algorithms targeting a reduced area controller. In order to bridge this gap a tool has been developed and integrated into `Workcraft` [2] which was named `SCENCO` (from **SC**enarios **ENCO**ding). As described in [13], an interesting option provided by this tool is the possibility to select custom encodings either by fixing the number of bits the designer wants the op-codes to be composed by, and by setting some particular bits. Both the two options might be applied to particular applications where some particular bits should be fixed in order to supply particular functions. This research may be useful to those companies which already design ASIP (Application Specific Instruction-set Processor) such as `Altera` with *Nios*, `Xilinx` with *Microblaze* or `Cadence` (`Tensilica` section) with *XTensa*. The model, that now features the technique presented in this paper for the encoding of the various scenarios, may support designing process of ASIP by reducing area of the final circuit and time to market.

There are many research directions come up with after this research. One of the most worthwhile to inspect is surely the seeking of a new cost function in order to improve the search of an optimal encoding even more [18]. As one might observe on Figure 3-b, the cost function so far is affected by a certain degree of error, which may also affect the result in terms of area of the encoding found. Additionally, synthesis of final controller starting from *Conditional Partial Order Graphs* might be driven by different cost parameters such as the delay, or the power consumption [19]. Furthermore, as CPOGs tailors perfectly to asynchronous controller design [15], a further research direction might be to integrate analogue specifications into the representation in order to represent an

asynchronous circuit able to satisfy the analogue requirements.

Acknowledgements: We would like to thank our supervisor Dr Andrey Mokhov for supporting us and this research inside the μSystems Group at Newcastle University. He has been instrumental for our work.

References

[1] Andrey Mokhov, Alex Yakovlev. *"Conditional partial order graphs and dynamically reconfigurable control synthesis"*. Design, Automation and Test in Europe, 2008. DATE'08, Pages 1142-1147, 10/03/2008.

[2] Ivan Poliakov, Danil Sokolov, Andrey Mokhov. *"Workcraft: A static data flow structure editing, visualisation and analysis tool"*. Petri Nets and Other Models of Concurrency - ICATPN 2007. Pages 505-514, 2007.

[3] ARM Ltd. *"ARMv6-M Architecture Reference Manual"*. ARM DDI 0419C (ID092410), 2010.

[4] H. Iwai, *"Roadmap for 22nm and beyond (Invited Paper)"*, Microelectronic Engineering, vol. 86, pp. 1520-1528, 7/2009.

[5] J. Rabaey, *"Low Power Design Essentials: New York, London"*; Springer 2009.

[6] Andrey Mokhov, Alex Yakovlev. *"Conditional partial order graphs: Model, synthesis, and application"*. IEEE Transactions on Computers, Volume 59, Pages 1480-1493, November 2010.

[7] Andrey Mokhov. *"Conditional Partial Order Graphs"*. Newcastle University, Ph.D. Thesis, Technical Report Series NCL-EECE-MSD-TR-2009-150, September 2009.

[8] S. Kirkpatrick, C. D. Gelatt Jr and M. P. Vecchi, *"Optimization by Simulated Annealing"*, vol. 220, number 4598, 13 May 1983.

[9] E. L. Lawlor, *"Combinatorial Optimization"*, (Holt, Rinehart & Winston, New York, 1976).

[10] A. V. Aho, J. E. Hopcroft, J. D. Ullman, *"The Design and Analysis of Computer Algorithms"*, (Addison-Wesley, Reading. Mass. 1974).

[11] Francisco Javier Rodriguez-Diaz, Carlos Garcia-Martinez and Manuel Lozano, *"A GA-Based Multiple Simulated Annealing"*, IEEE, Evolutionary Computation (CEC), July 2010.

[12] QI Ji-Yang, *"Application of Improved Simulated Annealing Algorithm in Facility Layout Design"*, IEEE, Proceedings of the 29th Chinese Control Conference, July 2010.

[13] Alessandro de Gennaro, *"Design of Reconfigurable Dataflow Processors"*. Technical Report NCL-EEE-MICRO-TR-2014-193, Master's Degree in Computer Engineering (Facolta' di Ingegneria, Politecnico Di Torino), μSystems Research Group, School of EEE, Newcastle University, October 2014.

[14] Paulius Stankaitis, *"Algebraic Specifications of ARM Cortex M0+ Instruction Set"*, Chapter 4. Technical report series NCL-EEE-MICRO-TR-2014-192, 2014.

[15] Danil Sokolov, Andrey Mokhov, Alex Yakovlev, David Lloyd. *"Towards asynchronous power management"*. 2014 IEEE Faible Tension Faible Consommation (FTFC), Pages 1-4, May 2014.

[16] Maxim Rykunov. *"Design of Asynchronous Microprocessor for Power Proportionality"*. Newcastle University, Ph.D. Thesis, December 2013.

[17] A Mokhov, A Alekseyev, A Yakovlev. *"Encoding of processor instruction sets with explicit concurrency control"*. Computers & Digital Techniques, IET. Volume 5, Pages 427-439. November 2011.

[18] Tiziano Villa. *"Encoding Problems in Logic Synthesis"*. University of California at Berkeley, Ph.D. Thesis in Engineering. 1995

[19] Andrey Mokhov, Maxim Rykunov, Danil Sokolov and Alex Yakovlev. *"Design of Processors with Reconfigurable Microarchitecture"*. J. Low Power Electron. Appl. 2014, 4, Pages 26-43. 20 January 2014.